Computer Physics Communications 184 (2013) 102–108

Contents lists available at SciVerse ScienceDirect

Computer Physics Communications

journal homepage: www.elsevier.com/locate/cpc

Efficient computation of Hamiltonian matrix elements between non-orthogonal Slater determinants

Yutaka Utsuno [a,b,*], Noritaka Shimizu [b], Takaharu Otsuka [b,c,d], Takashi Abe [b]

[a] Advanced Science Research Center, Japan Atomic Energy Agency, Tokai, Ibaraki 319-1195, Japan

[b] Center for Nuclear Study, University of Tokyo, Hongo Tokyo 113-0033, Japan

[c] Department of Physics, University of Tokyo, Hongo, Tokyo 113-0033, Japan

[d] National Superconducting Cyclotron Laboratory, Michigan State University, East Lansing, MI 48824, USA

ARTICLE INFO

Article history:
Received 28 December 2011
Received in revised form
31 July 2012
Accepted 2 September 2012
Available online 5 September 2012

Keywords:
Quantum many-body problem
Hamiltonian overlap
BLAS

ABSTRACT

We present an efficient numerical method for computing Hamiltonian matrix elements between non-orthogonal Slater determinants, focusing on the most time-consuming component of the calculation that involves a sparse array. In the usual case where many matrix elements should be calculated, this computation can be transformed into a multiplication of dense matrices. It is demonstrated that the present method based on the matrix–matrix multiplication attains ∼80% of the theoretical peak performance measured on systems equipped with modern microprocessors, a factor of 5–10 better than the normal method using indirectly indexed arrays to treat a sparse array. The reason for such different performances is discussed from the viewpoint of memory access.

1. Introduction

One of the main issues in the quantum many-body problem is solving a Schrödinger equation to good accuracy in reasonable computational time. While mean-field methods such as the Hartree–Fock method are very successful in various systems, the inclusion of effects beyond the mean field, i.e., correlation, is highly desired for better description. For instance, the mean-field wave function does not necessarily have a good quantum number that is conserved in the exact solution such as the total angular momentum.

A superposition of Slater determinants is the usual way to overcome the limitation of the mean-field method. Among various schemes to represent a correlated wave function, a representation by non-orthogonal Slater determinants (or quasiparticle vacuum states in general) is a method which is widely used in the nuclear many-body problems [1]. This method, often associated with the generator coordinate method (GCM) [2], has been successfully applied, for instance, to the description of collective motion and to the restoration of broken symmetry [3]. Recently, global studies of the correlation energy and the energy spectra over the nuclear chart have been carried out with the use of the GCM, for instance in [4–6]. Furthermore, the use of non-orthogonal Slater determinants has recently opened a new possibility for representing a precise many-body wave function in an efficient way, as demonstrated by the Monte Carlo shell model (MCSM) [7], variants of the VAMPIR method [8], and a hybrid method between MCSM and VAMPIR [9]. The MCSM method is now capable of precisely evaluating the eigenvalues even for a system beyond exact calculation by introducing a novel extrapolation method utilizing the variance of energy [10]. There have been some studies using the superposition of non-orthogonal Slater determinants also in quantum chemistry [11–14].

In the present paper, in order to extend the applicability of the expression of non-orthogonal Slater determinants, we present a numerical method for efficiently computing Hamiltonian matrix elements between them. Since we assume a general two-body force that has the rotational symmetry only, the present method will be applicable to various systems. This paper is organized as follows. Section 2 briefly describes the many-body system and many-body wave function under consideration. Section 3 presents some numerical methods for computing the most time-consuming part. In Section 4, the computational performances of the presented methods are compared, and the reason for their differences in performance is discussed. In Section 5, we summarize this paper.

* Corresponding author at: Advanced Science Research Center, Japan Atomic Energy Agency, Tokai, Ibaraki 319-1195, Japan. Tel.: +81 29 282 6901; fax: +81 29 282 5927.

E-mail address: utsuno.yutaka@jaea.go.jp (Y. Utsuno).

0010-4655/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.cpc.2012.09.002

2. Many-body calculation with non-orthogonal Slater determinants

In this paper, we consider the many-body system described by the Hamiltonian consisting of a one-body operator T and a two-body operator V,

$$H = T + V = \sum_{l_1 l_2}^{N_s} t_{l_1 l_2} c_{l_1}^\dagger c_{l_2} + \frac{1}{4} \sum_{l_1 l_2 l_3 l_4}^{N_s} \bar{v}_{l_1 l_2, l_3 l_4} c_{l_1}^\dagger c_{l_2}^\dagger c_{l_4} c_{l_3}, \tag{1}$$

where $c_l^\dagger$ and c_l are the creation and annihilation operators of the state labeled by l, respectively. The one-body matrix elements $t_{l_1 l_2}$ are given by $t_{l_1 l_2} = \langle l_1 | T | l_2 \rangle$, and the two-body matrix elements defined by $\bar{v}_{l_1 l_2, l_3 l_4} = \langle l_1 l_2 | V | l_3 l_4 \rangle - \langle l_1 l_2 | V | l_4 l_3 \rangle$ are antisymmetrized: $\bar{v}_{l_1 l_2, l_3 l_4} = -\bar{v}_{l_1 l_2, l_4 l_3}$. We consider a model space consisting of a finite number of single-particle orbits represented by N_s, and regard a set of the single-particle wave functions $\phi_l(x) = \langle x | c_l^\dagger | - \rangle$ ($l = 1, 2, \ldots, N_s$) as a *single-particle* basis set.

We approximate the solution of Eq. (1) by a superposition of a finite number of non-orthogonal Slater determinants

$$|\Psi\rangle = \sum_q f(q) |\Phi(q)\rangle, \tag{2}$$

where $|\Phi(q)\rangle$ and $f(q)$ denote a Slater determinant and its amplitude, respectively. Note that although the wave function $|\Psi\rangle$ is sometimes expressed by a continuous superposition over q as is expressed by the GCM, the actual numerical calculation is usually performed by the discretization shown in Eq. (2). Each Slater determinant, regarded as a *many-body* basis state, is represented by a product of generalized creation operators

$$|\Phi(q)\rangle = \prod_{i=1}^{N_p} a_i^\dagger(q) |-\rangle, \tag{3}$$

where N_p is the number of particles, and the creation operator $a_i^\dagger(q)$ is given by

$$a_i^\dagger(q) = \sum_l^{N_s} D(q)_{li} c_l^\dagger. \tag{4}$$

Here the $N_s \times N_p$ matrix ($N_s \geq N_p$) $D(q)$ characterizes the many-body basis state $|\Phi(q)\rangle$. In general, the basis states $|\Phi(q)\rangle$ are non-orthogonal between one another: $\langle \Phi(q') | \Phi(q) \rangle \neq 0$. Although an important issue in quantum many-body theory is how to choose good $|\Phi(q)\rangle$, we do not mention it here because the aim of this paper is to present an efficient computational method which is valid for any calculation of the same type. Once a set of the many-body basis states is fixed, one needs to optimize a set of amplitudes $f(q)$. This optimization is usually carried out with the variational principle:

$$\delta \frac{\langle \Psi | H | \Psi \rangle}{\langle \Psi | \Psi \rangle} = 0, \tag{5}$$

which leads to the Hill–Wheeler equation [2] for a discretized coordinate q:

$$\mathcal{H} f = E \mathcal{N} f, \tag{6}$$

where $\mathcal{H}$ and $\mathcal{N}$ are matrices whose elements are given by

$$\mathcal{H}(q', q) = \langle \Phi(q') | H | \Phi(q) \rangle \tag{7}$$

$$\mathcal{N}(q', q) = \langle \Phi(q') | \Phi(q) \rangle. \tag{8}$$

f is a vector whose component is $f(q)$, and E is the eigenvalue. Following the terminology of the GCM, we hereafter call the *many-body* matrix elements of $\mathcal{H}$ and $\mathcal{N}$ the Hamiltonian overlap and the norm overlap, respectively, to avoid confusing them with the two-body matrix element $\bar{v}_{l_1 l_2, l_3 l_4}$ of a single-particle basis. Both the overlaps are represented by $D(q)$ and $D(q')$. The norm overlap is written as

$$\mathcal{N}(q', q) = \det \left(D(q')^\dagger D(q) \right), \tag{9}$$

and the Hamiltonian overlap is

$$\mathcal{H}(q', q) = \mathcal{N}(q', q) \left(\sum_{l_1 l_2}^{N_s} t_{l_1 l_2} \rho_{l_2 l_1} + \frac{1}{2} \sum_{l_1 l_2 l_3 l_4}^{N_s} \rho_{l_3 l_1} \bar{v}_{l_1 l_2, l_3 l_4} \rho_{l_4 l_2} \right) \tag{10}$$

using the density matrix ρ whose matrix element is defined by

$$\rho_{ll'} = \frac{\langle \Phi(q') | c_{l'}^\dagger c_l | \Phi(q) \rangle}{\langle \Phi(q') | \Phi(q) \rangle}. \tag{11}$$

Using $D(q)$ and $D(q')$, the density matrix becomes

$$\rho = D(q) \left(D(q')^\dagger D(q) \right)^{-1} D(q')^\dagger. \tag{12}$$

The derivation of Eqs. (9), (10) and (12) is given in Appendix.

Among various applications of the above expression is the restoration of broken symmetries. Since a general Slater determinant of Eq. (3) does not necessarily possess the symmetries that the original Hamiltonian has, it is desirable to restore the broken symmetries by projecting the wave function onto good quantum numbers. The total angular momentum, for instance, is restored from $|\Phi\rangle$ by performing a three-dimensional integration over the Euler angles [1]. To carry out a numerical integration, the number of mesh points for the Euler angles is required to be as many as the order of 10^4, as are the numbers of $\mathcal{H}(q', q)$ and $\mathcal{N}(q', q)$ to be calculated [1].

As thus exemplified, innumerable Slater determinants are often involved to obtain a good many-body wave function $|\Psi\rangle$. Hence, fast computation of the Hamiltonian and norm overlaps will accelerate the whole calculation. The most time-consuming in the above procedure is the computation of the two-body part of the Hamiltonian overlap

$$\langle V \rangle \equiv \sum_{l_1 l_2 l_3 l_4}^{N_s} \rho_{l_3 l_1} \, \bar{v}_{l_1 l_2, l_3 l_4} \, \rho_{l_4 l_2} = \sum_{l_1 l_3}^{N_s} \rho_{l_3 l_1} \, \Gamma_{l_1 l_3}, \tag{13}$$

with

$$\Gamma_{kk'} = \sum_{ll'}^{N_s} \bar{v}_{kl', k'l} \, \rho_{ll'}, \tag{14}$$

because such computation requires a fourfold summation over the single-particle states. In the following sections, we concentrate on an efficient computational method for Eq. (13) on systems equipped with modern microprocessors. We assume that the operation of Eq. (13) is repeated a great number of times for different density matrices ρ under the condition of fixed two-body matrix elements $\bar{v}_{l_1 l_2, l_3 l_4}$.

3. Numerical methods for computing the Hamiltonian overlap

A straightforward operation of Eq. (13) is in general a waste of computational time because $\bar{v}_{l_1 l_2, l_3 l_4}$ is very sparse. This sparseness is due to the symmetry of the Hamiltonian. For instance, the conservation of the z component of the angular momentum leads to $\bar{v}_{l_1 l_2, l_3 l_4} = 0$ unless $j_z(l_1) + j_z(l_2) = j_z(l_3) + j_z(l_4)$ is satisfied.

Depending on the system considered, some other symmetries such as parity, orbital angular momentum, and isospin quantum numbers are also conserved, which imposes further constraints on the non-zero matrix elements. Hence, every effort must be made to avoid taking those vanishing matrix elements for efficient computing. Below we show three numerical algorithms for this purpose. The first method is completely different from the other two, and the last method is more advanced than the second method.

Indirect-index method

As shown in the last paragraph, the operation associated with zero for calculating $\langle V\rangle$ is mainly caused not by the density matrix but by the fixed two-body matrix elements $\bar{v}_{l_1l_2,l_3l_4}$. Thus, it is useful to classify in advance the indices (l_1, l_2, l_3, l_4) of $\bar{v}_{l_1l_2,l_3l_4}$ according to whether they lead to non-vanishing $\bar{v}_{l_1l_2,l_3l_4}$, and to label the set of indices (l_1, l_2, l_3, l_4) satisfying this condition with a so-called indirect index k as $(l_1(k), l_2(k), l_3(k), l_4(k))$. Eq. (13) is then represented as

$$\langle V\rangle = \sum_{k}^{N_{\rm nonzero}} \rho_{l_3(k)l_1(k)}\, \bar{v}_{\rm nonzero}(k)\, \rho_{l_4(k)l_2(k)}, \tag{15}$$

where $N_{\rm nonzero}$ is the number of non-vanishing $\bar{v}_{l_1l_2,l_3l_4}$, and $\bar{v}_{\rm nonzero}(k) \equiv \bar{v}_{l_1(k)l_2(k),l_3(k)l_4(k)} \neq 0$. When $\bar{v}_{l_1l_2,l_3l_4}$ is sparse, $N_{\rm nonzero}$ is much smaller than N_s^4. In this paper, we refer to the numerical algorithm based on Eq. (15) as the *indirect-index method*.

Matrix–vector method

Although the introduction of the indirect index can always be applied to the computation of sparse arrays, here we present an alternative numerical approach which directly utilizes the symmetry. We now assume that the two-body force V has only the rotational invariance for simplicity. Other possible symmetries can be treated in a similar way.

First, $N_s \times N_s$ density-matrix elements $\rho_{ll'}$ are grouped according to $\Delta m \equiv j_z(l') - j_z(l)$, and the set of (l, l') having a common Δm is indexed by $k = 1, 2, \ldots, N_{\Delta m}$ as $\tilde{\rho}(\Delta m)_k$. In a similar way, the two-body matrix elements $\bar{v}_{l_1l_2,l_3l_4}$ are categorized according to $\Delta m_{13} \equiv j_z(l_1) - j_z(l_3)$ and $\Delta m_{24} \equiv j_z(l_2) - j_z(l_4)$ as $\tilde{v}(\Delta m_{13}, \Delta m_{24})_{k'k}$, where k' and k are, respectively, indices to (l_1, l_3) and (l_2, l_4) having Δm_{13} and Δm_{24}. Eq. (13) then leads to

$$\begin{aligned}\langle V\rangle &= \sum_{\Delta m_{13}\Delta m_{24}} \sum_{k'k} \tilde{\rho}(\Delta m_{13})_{k'}\, \tilde{v}(\Delta m_{13}, \Delta m_{24})_{k'k} \tilde{\rho}(\Delta m_{24})_k \\ &= \sum_{\Delta m} \sum_{k'k} \tilde{\rho}(-\Delta m)_{k'}\, \tilde{v}(-\Delta m, \Delta m)_{k'k} \tilde{\rho}(\Delta m)_k,\end{aligned} \tag{16}$$

where the last equation of Eq. (16) is derived from the necessary condition for $\bar{v}_{l_1l_2,l_3l_4}$ being non-zero: $j_z(l_1)+j_z(l_2) = j_z(l_3)+j_z(l_4)$, i.e., $\Delta m_{13} = j_z(l_1) - j_z(l_3) = -(j_z(l_2) - j_z(l_4)) = -\Delta m_{24} \equiv -\Delta m$.

Since the density matrix $\tilde{\rho}(\Delta m)$ and the two-body matrix $\tilde{v}(-\Delta m, \Delta m)$ for a given Δm are a one-dimensional array and a two-dimensional array, respectively, they can be identified with a vector of size $N_{\Delta m}$ and a matrix of size $N_{\Delta m} \times N_{\Delta m}$, respectively, by using $N_{\Delta m} = N_{-\Delta m}$. Thus, Eq. (16) is regarded as a ^{t}(vector) $\times$ (matrix) $\times$ (vector) operation. This is schematically illustrated in Fig. 1. It is clearly seen that the sparse array $\bar{v}_{l_1l_2,l_3l_4}$ is transformed into a block-antidiagonal matrix $\tilde{v}$ whose blocks are dense submatrices. In this paper, we refer to the numerical algorithm based on Eq. (16) as the *matrix–vector method*.

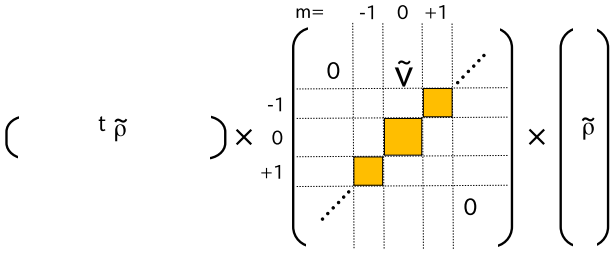

Fig. 1. Schematic illustration of the operation of Eq. (16).

Matrix–matrix method

In the matrix–vector method, most of the computational time is devoted to the (matrix) $\times$ (vector) operation $\tilde{\Gamma} \equiv \tilde{v}\tilde{\rho}$, where the index of Δm is omitted for simplicity. As previously mentioned, this operation is usually repeated a number of times for different $\tilde{\rho}$'s: $\tilde{v}\tilde{\rho}^{(1)}, \tilde{v}\tilde{\rho}^{(2)}, \ldots$. By binding vectors $\tilde{\rho}^{(1)}, \tilde{\rho}^{(2)}, \ldots, \tilde{\rho}^{(N_{\rm vec})}$ into a matrix $\theta \equiv (\tilde{\rho}^{(1)}, \tilde{\rho}^{(2)}, \ldots, \tilde{\rho}^{(N_{\rm vec})})$, repeated (matrix) $\times$ (vector) operations are performed by a (matrix) $\times$ (matrix) operation at one time:

$$\begin{aligned}(\tilde{\Gamma}^{(1)}, \tilde{\Gamma}^{(2)}, \ldots, \tilde{\Gamma}^{(N_{\rm vec})}) &= (\tilde{v}\tilde{\rho}^{(1)}, \tilde{v}\tilde{\rho}^{(2)}, \ldots, \tilde{v}\tilde{\rho}^{(N_{\rm vec})}) \\ &= \tilde{v}\theta,\end{aligned} \tag{17}$$

where the number of columns $N_{\rm vec}$ can be chosen arbitrarily. The $\langle V\rangle$ for the i-th density matrix $\tilde{\rho}^{(i)}$ is then given by ${}^t\tilde{\rho}^{(i)}\tilde{v}\tilde{\rho}^{(i)} = {}^t\tilde{\rho}^{(i)}\tilde{\Gamma}^{(i)} = {}^t\tilde{\rho}^{(i)}(\tilde{v}\theta)^{(i)}$, where $(\tilde{v}\theta)^{(i)}$ stands for the i-th column of the matrix $\tilde{v}\theta$. We call this method, i.e., the way through the (matrix) $\times$ (matrix) operation of Eq. (17), the *matrix–matrix method*. It seems as if there is no substantial difference between the matrix–matrix method and the matrix–vector method: Eq. (17) keeps not only mathematical identity but also the number of elementary operations. However, as seen in the next section, those two methods result in quite different computational performances on actual computer systems.

Case of the quasiparticle vacuum state

Although this paper concentrates on the Hamiltonian overlap between Slater determinants, it is useful to mention applicability to the Hamiltonian overlap between quasiparticle vacuum states. The quasiparticle vacuum state is a generalized single-particle state, and is widely used in nuclear physics to include the pairing correlation. Similar to Eq. (10), the Hamiltonian overlap for the quasiparticle vacuum state is written [1] as

$$\begin{aligned}\langle\Phi(q')|H|\Phi(q)\rangle = \langle\Phi(q')|\Phi(q)\rangle \Bigg(&\sum_{l_1l_2}^{N_s} t_{l_1l_2}\rho^{10}_{l_2l_1} \\ &+ \frac{1}{2}\sum_{l_1l_2l_3l_4}^{N_s} \rho^{10}_{l_3l_1}\bar{v}_{l_1l_2,l_3l_4}\rho^{10}_{l_4l_2} \\ &+ \frac{1}{4}\sum_{l_1l_2l_3l_4}^{N_s} \kappa^{01*}_{l_1l_2}\bar{v}_{l_1l_2,l_3l_4}\kappa^{10}_{l_3l_4}\Bigg),\end{aligned} \tag{18}$$

where the density matrix ρ^{10} and the pairing tensors κ^{10} and κ^{01*} are defined by

$$\rho^{10}_{ll'} = \frac{\langle\Phi(q')|c^\dagger_{l'}c_l|\Phi(q)\rangle}{\langle\Phi(q')|\Phi(q)\rangle} \tag{19}$$

$$\kappa^{10}_{ll'} = \frac{\langle\Phi(q')|c_{l'}c_l|\Phi(q)\rangle}{\langle\Phi(q')|\Phi(q)\rangle} \tag{20}$$

$$\kappa^{01*}_{ll'} = \frac{\langle\Phi(q')|c^\dagger_l c^\dagger_{l'}|\Phi(q)\rangle}{\langle\Phi(q')|\Phi(q)\rangle}. \tag{21}$$

The difference from the Slater determinant is the addition of the last term in Eq. (18). Its computation with the matrix–vector or matrix–matrix method is rather similar. The two-body matrix elements $\bar{v}_{l_1l_2,l_3l_4}$ are categorized according to $M_{12} \equiv j_z(l_1) + j_z(l_2)$ and $M_{34} \equiv j_z(l_3) + j_z(l_4)$ as $\tilde{w}(M_{12}, M_{34})_{k'k}$, where k' and k are indices to (l_1, l_2) having M_{12} and (l_3, l_4) having M_{34}, respectively. Since the necessary condition for $\bar{v}_{l_1l_2,l_3l_4}$ being non-zero is $M_{12} = M_{34}$ ($\equiv M$), the two-body matrix elements are block diagonalized as $\tilde{w}(M, M)_{k'k}$ each of which is a dense matrix. The pairing tensor can be regarded as a vector $\tilde{\kappa}$ in this representation. Thus, when the matrix–vector or matrix–matrix method is applied to the quasiparticle vacuum state, one needs to prepare two kinds of matrix representations of $\bar{v}_{l_1l_2,l_3l_4}$, $\tilde{v}$ and $\tilde{w}$, the former and the latter of which act on the vector representation of the density matrix, $\tilde{\rho}$, and the vector representation of the pairing tensor, $\tilde{\kappa}$, respectively.

4. Measurement of performance

In this section, computational performance is compared among the three methods presented in the last section by adopting a realistic many-body system and measuring the elapsed time to compute Eq. (13) repeatedly.

4.1. Benchmark system

Here we consider a nuclear many-body problem where protons and neutrons interact in a fixed model space. We adopt a set of the single-particle orbits consisting of five harmonic-oscillator major shells from harmonic-oscillator's quantum number $N_{\rm osc} = 0$ to 4: $0s_{1/2}$, $0p_{3/2}$, $0p_{1/2}$, $0d_{5/2}$, $0d_{3/2}$, $1s_{1/2}$, $0f_{7/2}$, $0f_{5/2}$, $1p_{3/2}$, $1p_{1/2}$, $0g_{9/2}$, $0g_{7/2}$, $1d_{5/2}$, $1d_{3/2}$, and $2s_{1/2}$. Thus, the number of the proton (neutron) single-particle states N_s is 70. Here, the proton and neutron numbers are set to be two and two, respectively, but the number of particles is irrelevant to the computational time of Eq. (13).

The two-body part of the adopted Hamiltonian is an arbitrary one that has rotational, parity and time-reversal symmetries. Due to the rotational and time-reversal symmetries, all the matrix elements $\bar{v}_{l_1l_2,l_3l_4}$ can be real numbers [15]. Since we do not assume other symmetries such as an isospin, we calculate the proton–proton interaction part of Eq. (13), the neutron-neutron part, and the proton–neutron part independently. For this system, the largest submatrix used in the matrix–vector (or matrix–matrix) method is of the size 390×390, classified according to the z component of the angular momentum and the parity.

The wave function taken is a single Slater determinant with total angular-momentum and parity projection. Each single-particle state of the Slater determinant is assumed to be a pure proton or neutron state. The number of mesh points for the three Euler angles and that for the parity projector are 25^3 and 2, respectively, leading to $25^3 \times 2 = 31,250$ times the computations of Eq. (13). Since a rotation of a wave function involves imaginary numbers [1], the density matrix has to be complex.

It would be useful to compare the number of elementary floating-point operations (addition and multiplication) among the three methods. Taking into account that the loop length of Eq. (13) can be halved by using $\bar{v}_{l_1l_2,l_3l_4} = \bar{v}_{l_2l_1,l_4l_3}$, the number of elementary floating-point operations becomes 20,992,518 for the indirect-index method and 10,365,224 for the matrix–vector method and the matrix–matrix method. The former is almost the double of the latter as explained as follows. In the matrix–vector method, $(\tilde{v}\tilde{\rho})_{k'} = \tilde{v}_{k'1}\tilde{\rho}_1 + \tilde{v}_{k'2}\tilde{\rho}_2 + \cdots$ is factored out of $\sum_{k'k} \tilde{\rho}_{k'}\tilde{v}_{k'k}\tilde{\rho}_k$ in the way $\sum_{k'} \tilde{\rho}_{k'}(\tilde{v}_{k'1}\tilde{\rho}_1 + \tilde{v}_{k'2}\tilde{\rho}_2 + \cdots)$. This expression saves the number of multiplications, and more importantly, the reduced operations are the multiplication of *complex numbers* which costs as many as six floating-point operations.

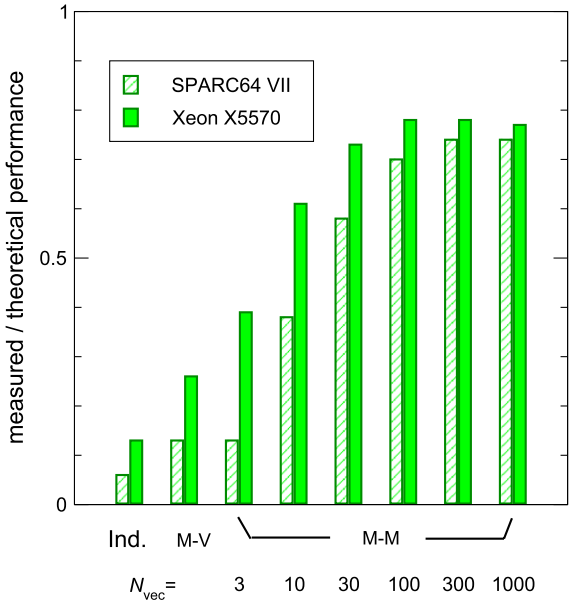

Fig. 2. Comparison of the computational performance among the indirect-index method (Ind.), matrix–vector method (M–V) and matrix–matrix method (M–M) with different $N_{\rm vec}$ measured on the SPARC64 VII and Xeon X5570 systems. The values are normalized by their theoretical peak performance. See the text for more details.

4.2. Computational environment

The computation is carried out as a single-threaded process on two different systems based on up-to-date scalar processors: one system is based on the Xeon X5570 processor with clock speed 2.93 GHz and the other is based on the SPARC64 VII processor with clock speed 2.5 GHz. Their theoretical peak performances per CPU core are 11.72 GFLOPS and 10 GFLOPS, respectively. Our code written in Fortran 90/95/2003 is compiled by Intel Fortran Compiler Version 11.1 for the Xeon system and by Fujitsu Fortran Compiler Driver Version 8.2 for the SPARC64 system. The two-body matrix elements $\bar{v}_{l_1l_2,l_3l_4}$ and the density matrix elements $\rho_{ll'}$ are of double-precision. Matrix and/or vector calculations are coded to call the BLAS interface (BLAS [16] is the de facto standard for the programming interface of basic linear algebra operations). We use optimized BLAS implementations: Intel Math Kernel Library (MKL) for the Xeon system and Fujitsu Scientific Subroutine Library II (SSL II) for the SPARC64 system. The computational performance for executing Eq. (13) is measured with the wall-clock time at a microsecond-level resolution, which is good enough for the present purpose.

4.3. Results and analyses

The performance of a computation is characterized by the inversion of the wall-clock time t. It is comprehensive to express the performance in FLOPS which is t^{-1} (in 1/s) multiplied by the total number of elementary floating-point operations executed. But since the number of operations is different among the methods as shown previously, FLOPS is not a good measure for comparing their relative performances. Hence, to make direct comparison possible, the performance is now defined by t^{-1} multiplied by a fixed factor of the number of elementary floating-point operations of the matrix–vector (or the matrix–matrix) method, only when it serves as the actual FLOPS.

Fig. 2 compares the measured performances normalized by the theoretical peak performances of the adopted systems. The indirect-index method gives the lowest performance for both

systems. The performance of the matrix–vector method is about twice as high as that of the indirect-index method. This is almost equivalent to the ratio of the numbers of floating-point operations, but is still far from the theoretical peak performance. When $\tilde{\rho}$ vectors are bound into a matrix in the matrix–matrix method, the performance starts to increase. The performance improves sharply even at a small $N_{\rm vec}$, and is saturated at around $N_{\rm vec} \sim 30$–100 to reach ∼70%–80% of the theoretical peak performance. The values of the two systems are very close at a large $N_{\rm vec}$ in contrast to rather different behavior at a smaller $N_{\rm vec}$.

Although the matrix–vector and matrix–matrix methods are identical in mathematics, they are quite different in performance. Memory access, the major bottleneck of modern computer systems, differentiates the methods from each other. Now we consider a matrix of size $n \times n$ and a vector of size n and estimate the number of arithmetic operations and memory accesses involving them. Since a matrix-times-vector operation needs $2n^2$ floating-point operations and $n^2 + n$ memory accesses, the computational intensity defined by their ratio is ∼2. On the other hand, the computational intensity for a matrix-times-matrix operation becomes n, much larger than that of the matrix–vector operation for a sufficiently large n. More specifically, the matrix-times-matrix operation can be designed so that most of the CPU time can be involved in arithmetic operations rather than memory access as is implemented in numerical libraries such as MKL and SSL II. See, for instance, [17] for more detailed analyses of the performance of basic linear algebra operations in terms of computer architecture.

We also consider the performance of parallel processes. We take an example where the 31,250 ${}^t\tilde{\rho}\tilde{v}\tilde{\rho}$ operations are divided into 32 MPI processes running on a 4 node × 2 CPU × 4 core Xeon X5570 system. The matrix–matrix method with $N_{\rm vec} = 100$ reaches 8.5 GFLOPS/core, which is rather close to the 9.1 GFLOPS achieved by the single process. In contrast, for the matrix–vector method, the parallel performance is reduced to 1.5 GFLOPS/core from the single-process performance of 3.1 GFLOPS. This difference is also accounted for by the memory access: since the memory bandwidth is shared by all the CPU cores on the board, the effective bandwidth defined by the bandwidth per process or thread is reduced for parallel processes. This reduction of the effective bandwidth leads to the reduction of the performance particularly for the processes involving heavy memory access like the matrix–vector operation. Thus, the matrix–matrix method is superior to the matrix–vector method not only in absolute performance but also in parallel efficiency because of less memory demanding formalism.

4.4. Towards larger calculations

The benchmark calculation is carried out for the model space $N_{\rm shell} = 5$, where $N_{\rm shell}$ stands for the number of harmonic oscillator major shells included. Although this model space is large for the shell-model calculation, it is not large enough for the density-functional calculation. Hence, in view of possible application to modern density-functional calculations, we examine how computational requirements change as the model space is enlarged.

Fig. 3 shows the increase of $N_{\rm op}$ defined as the number of elementary floating-point operations for computing a single $\langle V \rangle$ of Eq. (13) with the matrix–vector method. In estimating the number, it is assumed that a multiplication and an addition of two complex numbers need six and two floating-point operations, respectively, while a multiplication between a real number and a complex number needs two floating-point operations. Fig. 3 indicates that $N_{\rm op}$ increases roughly exponentially with $N_{\rm shell}$ but that the slope decreases. As a result, the computational time for $N_{\rm shell} = 10$ is, for instance, $\sim 10^3$ times larger than the one for $N_{\rm shell} = 5$ if the effective performance shown in Fig. 2 is unchanged. This

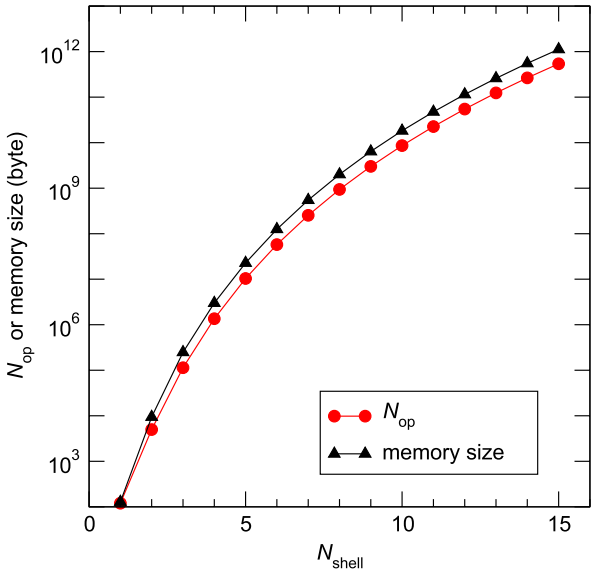

Fig. 3. The number of elementary floating-point operations $N_{\rm op}$ and memory size in bytes needed in the matrix–vector method as a function of $N_{\rm shell}$. Double precision data are assumed for estimating the memory size.

assumption is reasonable because the effective performance for a multiplication of large matrices is known to be kept high. Indeed, we have confirmed that almost the same performance is obtained for $N_{\rm shell} = 6$.

On the practical side, memory size could be a problem. As demonstrated in Fig. 3, gigabytes of memory are required to store the two-body matrix elements $\bar{v}_{l_1 l_2, l_3 l_4}$ for $N_{\rm shell} \geq 8$. However, this restriction due to the memory size can be relaxed when matrices $\tilde{v}(\Delta m, \Delta m)$ having different Δm are distributed over different nodes. In the case of $N_{\rm shell} = 10$, since the largest matrix size is 12,444, the maximum memory size is reduced to ∼1 GB. Finally, it should be noted that while the memory size needed for the calculation of multiple $\langle V \rangle$'s using the matrix–matrix method is almost unchanged from that of single $\langle V \rangle$, the number of floating-point operations is multiplied by $N_{\rm vec}$, i.e., the number of vectors bound (see Eq. (17)). Thus, the ratio of the number of memory access to the number of operations decreases accordingly, as discussed in Section 4.3.

5. Summary

We have presented an efficient numerical method for computing Hamiltonian matrix elements between non-orthogonal Slater determinants, motivated by recent findings that a superposition of non-orthogonal Slater determinants is a very effective way to solve a many-body problem. The most computationally demanding is the computation of a four-fold loop $\langle V \rangle = \sum_{l_1 l_2 l_3 l_4} \rho_{l_3 l_1} \bar{v}_{l_1 l_2, l_3 l_4} \rho_{l_4 l_2}$, where $\bar{v}_{l_1 l_2, l_3 l_4}$ is a sparse array due to the symmetries of the Hamiltonian. While indirectly indexed arrays are often introduced for treating a sparse matrix, the performance of the method has been measured to be much lower than the theoretical peak performance. In order to fit a formula of calculating $\langle V \rangle$ to fast computation, its key part is transformed into a multiplication of a dense matrix and a vector for a single $\langle V \rangle$ calculation. This formula is also transformed into a multiplication of dense matrices for multiple $\langle V \rangle$ calculations. The method based on the matrix–matrix multiplication attains as much as ∼80% of the theoretical peak performance on actual systems. Its high performance is accounted for by its high computational intensity, i.e., a large ratio of floating-point operations to memory accesses. Since from the hardware side it is predicted that the Byte/FLOP rate of future systems will be decreased [18] because of rapid increase of the

number of CPU cores compared to memory bandwidth, numerical methods should be developed so that the computational intensity can be higher as achieved by the present method.

Acknowledgments

One of the authors (Y.U.) thanks Prof. J. Dobaczewski for fruitful discussions during his stay at the European Centre for Theoretical Studies in Nuclear Physics and Related Areas (ECT*). This work was in part supported by MEXT Grant-in-Aid for Young Scientists (B) (21740204), for Scientific Research on Innovative Areas (No. 20105003), for Scientific Research (A) (20244022, 23244049), and the HPCI Strategic Program of MEXT. This work is a part of the RIKEN-CNS joint research project on large-scale nuclear-structure calculations. The numerical calculation was carried out on the BX900 and FX1 supercomputers at the Japan Atomic Energy Agency.

Appendix. Derivation of the formulae for calculating the norm and Hamiltonian overlaps

In this appendix, we derive the formulae for calculating the norm and Hamiltonian overlaps given by Eqs. (9) and (10). In the following, it is convenient to introduce unoccupied states

$$a_m^\dagger(q) = \sum_l^{N_s} \tilde{D}(q)_{lm} c_l^\dagger. \tag{A.1}$$

Hereafter, the occupied and unoccupied states are labeled by the indices i, j and m, n, respectively. Using Eqs. (4) and (A.1), the creation operator of a single-particle basis state $c_l^\dagger$ is written by $a_i^\dagger(q)$ and $a_m^\dagger(q)$ as

$$c_l^\dagger = \sum_i D(q)^*_{li} a_i^\dagger + \sum_m \tilde{D}(q)^*_{lm} a_m^\dagger. \tag{A.2}$$

The anticommutation relation $\{c_l, c_{l'}^\dagger\} = \delta_{ll'}$ leads to

$$D(q)D(q)^\dagger + \tilde{D}(q)\tilde{D}(q)^\dagger = I, \tag{A.3}$$

where I is the identity matrix. The creation operator $a_j^\dagger(q')$ can be expressed as a linear combination of $a_i^\dagger(q)$ and $a_m^\dagger(q)$ by using Eqs. (4) and (A.2):

$$a_j^\dagger(q') = \sum_i E_{ij} a_i^\dagger(q) + \sum_m \tilde{E}_{mj} a_m^\dagger(q), \tag{A.4}$$

where E and $\tilde{E}$ are given by $E = D(q)^\dagger D(q')$ and $\tilde{E} = \tilde{D}(q)^\dagger D(q')$, respectively.

A.1. Norm overlap

The overlap between $|\Phi(q')\rangle$ and $|\Phi(q)\rangle$ is calculated as

$$\begin{aligned}
&\langle\Phi(q')|\Phi(q)\rangle \\
&= \langle -| \prod_{i'=N_p}^{1} a_{i'}(q') \prod_{i=1}^{N_p} a_i^\dagger(q) |-\rangle \\
&= \langle -| \prod_{i'=N_p}^{1} \sum_j E^*_{ji'} a_j(q) \prod_{i=1}^{N_p} a_i^\dagger(q) |-\rangle \\
&= \sum_{\sigma\in S_{N_p}} E^*_{\sigma(1)1} \cdots E^*_{\sigma(N_p)N_p} \\
&\quad \times \langle -| a_{\sigma(N_p)} \cdots a_{\sigma(1)} a_1^\dagger \cdots a_{N_p}^\dagger |-\rangle \\
&= \det E^\dagger \\
&= \det\left(D(q')^\dagger D(q)\right),
\end{aligned} \tag{A.5}$$

where S_{N_p} stands for the symmetric group of degree N_p.

A.2. Hamiltonian overlap

According to Thouless' theorem [19], any Slater determinant $|\Phi(q')\rangle$ that is not orthogonal to a Slater determinant $|\Phi(q)\rangle$ can be expressed as

$$|\Phi(q')\rangle = N e^{\hat{Z}} |\Phi(q)\rangle, \tag{A.6}$$

where $\hat{Z} = \sum_{i,m} Z_{mi} a_m^\dagger a_i$. The normalization constant N is given by $N = \langle\Phi(q)|\Phi(q')\rangle$. Using Eq. (A.6), a general matrix element between $|\Phi(q')\rangle$ and $|\Phi(q)\rangle$ is

$$\begin{aligned}
&\langle\Phi(q')|c_{l_1}^\dagger \cdots c_{l_p}^\dagger c_{k_1} \cdots c_{k_q}|\Phi(q)\rangle \\
&= \langle\Phi(q')|\Phi(q)\rangle \langle\Phi(q)|\bar{d}_{l_1} \cdots \bar{d}_{l_p} d_{k_1} \cdots d_{k_q}|\Phi(q)\rangle,
\end{aligned} \tag{A.7}$$

where $\bar{d}_l$ and d_l are defined by $\bar{d}_l = e^{\hat{Z}^\dagger} c_l^\dagger e^{-\hat{Z}^\dagger}$ and $d_l = e^{\hat{Z}^\dagger} c_l e^{-\hat{Z}^\dagger}$, respectively. When the creation operator $b^\dagger$ is defined by $b_m^\dagger = a_m^\dagger$ and $b_i^\dagger = a_i$, $|\Phi(q)\rangle$ is regarded as vacuum:

$$b_l|\Phi(q)\rangle = 0 \tag{A.8}$$

for any single-particle state l. Hence, it is useful to represent $\bar{d}_l$ and d_l with $b^\dagger$ and b. Hereafter, $D(q)$ is simply written as D when no confusion is possible. Using the Baker–Hausdorff formula, it is straightforward to derive

$$\begin{aligned}
\bar{d}_l &= \sum_i \left(D^*_{li} + \sum_m Z^*_{mi} \tilde{D}^*_{lm} \right) b_i + \sum_m \tilde{D}^*_{lm} b_m^\dagger \\
d_l &= \sum_i D_{li} b_i^\dagger + \sum_m \left(\tilde{D}_{lm} - \sum_i Z^*_{mi} D_{li} \right) b_m.
\end{aligned} \tag{A.9}$$

Wick's theorem [20] is helpful to calculate the right hand side of Eq. (A.7). To use this theorem, the contraction of operators U and V defined as $U^\bullet V^\bullet = UV - :UV:$ is needed, where $:UV:$ stands for the normal ordered product of UV concerning $b^\dagger$ and b. Different expressions $U^{\bullet\bullet}V^{\bullet\bullet}$, $U^{\bullet\bullet\bullet}V^{\bullet\bullet\bullet}$ *etc.* are also used for the contraction $U^\bullet V^\bullet$ in order to specify the pair of operators considered. This definition leads to the following contractions

$$\begin{aligned}
\bar{d}_p^\bullet d_q^\bullet &= \left(D(D^\dagger + Z^\dagger \tilde{D}^\dagger) \right)_{qp} \\
d_p^\bullet \bar{d}_q^\bullet &= \left((\tilde{D} - DZ^\dagger)\tilde{D}^\dagger \right)_{pq} \\
\bar{d}_p^\bullet \bar{d}_q^\bullet &= 0 \\
d_p^\bullet d_q^\bullet &= 0.
\end{aligned} \tag{A.10}$$

The density matrix defined in Eq. (11) is identical with the contraction (see Eq. (A.7)):

$$\rho_{ll'} = \bar{d}_{l'}^\bullet d_l^\bullet = \left(D(D^\dagger + Z^\dagger \tilde{D}^\dagger) \right)_{ll'}. \tag{A.11}$$

The explicit form of the matrix Z can be derived from the condition

$$a_i^\dagger(q')|\Phi(q')\rangle = N e^{\hat{Z}} \left(e^{-\hat{Z}} a_i^\dagger(q') e^{\hat{Z}} \right) |\Phi(q)\rangle = 0. \tag{A.12}$$

After some lengthy calculations, it is proved that this is satisfied when Z is taken to be

$$Z = \tilde{E}E^{-1} = \tilde{D}(q)^\dagger D(q')(D(q)^\dagger D(q'))^{-1}. \tag{A.13}$$

Thus, the expression

$$\rho = D(q) \left(D(q')^\dagger D(q) \right)^{-1} D(q')^\dagger \tag{A.14}$$

is obtained by substituting Eq. (A.13) for Eq. (A.11) and using Eq. (A.3).

In the case of a two-body operator $\bar{d}_{l_1}\bar{d}_{l_2}d_{l_4}d_{l_3}$, Wick's theorem leads to

$$\begin{aligned}\bar{d}_{l_1}\bar{d}_{l_2}d_{l_4}d_{l_3} &= :\bar{d}_{l_1}\bar{d}_{l_2}d_{l_4}d_{l_3}: + :\bar{d}^{\bullet}_{l_1}\bar{d}^{\bullet}_{l_2}d_{l_4}d_{l_3}: \\ &\quad + :\bar{d}^{\bullet}_{l_1}\bar{d}_{l_2}d^{\bullet}_{l_4}d_{l_3}: + \cdots + :\bar{d}^{\bullet}_{l_1}\bar{d}^{\bullet}_{l_2}d^{\bullet\bullet}_{l_4}d^{\bullet\bullet}_{l_3}: \\ &\quad + :\bar{d}^{\bullet}_{l_1}\bar{d}^{\bullet\bullet}_{l_2}d^{\bullet}_{l_4}d^{\bullet\bullet}_{l_3}: + :\bar{d}^{\bullet}_{l_1}\bar{d}^{\bullet\bullet}_{l_2}d^{\bullet\bullet}_{l_4}d^{\bullet}_{l_3}: . \end{aligned} \tag{A.15}$$

While the last three terms of the right hand side of Eq. (A.15) are c-numbers, the other terms include b and/or $b^\dagger$ in the normal order and produce vanishing diagonal matrix elements for $|\Phi(q)\rangle$ because of Eq. (A.8). Thus, the general matrix element of a two-body operator $c^\dagger_{l_1}c^\dagger_{l_2}c_{l_4}c_{l_3}$ between $|\Phi(q)\rangle$ and $|\Phi(q')\rangle$ is given by Eq. (A.7):

$$\begin{aligned}&\langle\Phi(q')|c^\dagger_{l_1}c^\dagger_{l_2}c_{l_4}c_{l_3}|\Phi(q)\rangle \\ &\quad= \langle\Phi(q')|\Phi(q)\rangle\langle\Phi(q)|\bar{d}_{l_1}\bar{d}_{l_2}d_{l_4}d_{l_3}|\Phi(q)\rangle \\ &\quad= \langle\Phi(q')|\Phi(q)\rangle(:\bar{d}^{\bullet}_{l_1}\bar{d}^{\bullet}_{l_2}d^{\bullet\bullet}_{l_4}d^{\bullet\bullet}_{l_3}: \\ &\qquad + :\bar{d}^{\bullet}_{l_1}\bar{d}^{\bullet\bullet}_{l_2}d^{\bullet}_{l_4}d^{\bullet\bullet}_{l_3}: + :\bar{d}^{\bullet}_{l_1}\bar{d}^{\bullet\bullet}_{l_2}d^{\bullet\bullet}_{l_4}d^{\bullet}_{l_3}:) \\ &\quad= \langle\Phi(q')|\Phi(q)\rangle(\rho_{l_3l_1}\rho_{l_4l_2} - \rho_{l_4l_1}\rho_{l_3l_2}), \end{aligned} \tag{A.16}$$

where $:\bar{d}^{\bullet}_{l_1}\bar{d}^{\bullet}_{l_2}d^{\bullet\bullet}_{l_4}d^{\bullet\bullet}_{l_3}: = 0$, $:\bar{d}^{\bullet}_{l_1}\bar{d}^{\bullet\bullet}_{l_2}d^{\bullet}_{l_4}d^{\bullet\bullet}_{l_3}: = -:\bar{d}^{\bullet}_{l_1}d^{\bullet}_{l_4}\bar{d}^{\bullet\bullet}_{l_2}d^{\bullet\bullet}_{l_3}: = -\rho_{l_4l_1}\rho_{l_3l_2}$, and $:\bar{d}^{\bullet}_{l_1}\bar{d}^{\bullet\bullet}_{l_2}d^{\bullet\bullet}_{l_4}d^{\bullet}_{l_3}: =: \bar{d}^{\bullet}_{l_1}d^{\bullet}_{l_3}\bar{d}^{\bullet\bullet}_{l_2}d^{\bullet\bullet}_{l_4}: = \rho_{l_3l_1}\rho_{l_4l_2}$ are used. It is noted that any transposition of two operators in the contraction changes the sign (see Rule C″ in [20]). This gives Eq. (10) straightforwardly for antisymmetrized two-body matrix elements satisfying $\bar{v}_{l_1l_2,l_3l_4} = -\bar{v}_{l_1l_2,l_4l_3}$.

References

[1] For instance P. Ring, P. Schuck, The Nuclear Many-Body Problem, Springer, 1980.
[2] D.L. Hill, J.A. Wheeler, Phys. Rev. 89 (1953) 1102;
J.J. Griffin, J.A. Wheeler, Phys. Rev. 108 (1957) 311.
[3] M. Bender, P.-H. Heenen, P.-G. Reinhard, Rev. Modern Phys. 75 (2003) 121.
[4] M. Bender, G.F. Bertsch, P.-H. Heenen, Phys. Rev. C 73 (2006) 034322.
[5] B. Sabbey, M. Bender, G.F. Bertsch, P.-H. Heenen, Phys. Rev. C 75 (2007) 044305.
[6] L.M. Robledo, G.F. Bertsch, Phys. Rev. C 84 (2011) 054302.
[7] T. Otsuka, M. Honma, T. Mizusaki, N. Shimizu, Y. Utsuno, Prog. Part. Nucl. Phys. 47 (2001) 319.
[8] K.W. Schmid, Prog. Part. Nucl. Phys. 52 (2004) 565.
[9] G. Puddu, J. Phys. G 32 (2006) 321.
[10] N. Shimizu, Y. Utsuno, T. Mizusaki, T. Otsuka, T. Abe, M. Honma, Phys. Rev. C 82 (2010) 061305(R).
[11] H. Koch, E. Dalgaard, Chem. Phys. Lett. 212 (1993) 193.
[12] N. Tomita, S. Ten-no, Y. Tanimura, Chem. Phys. Lett. 263 (1996) 687.
[13] N.H. Morgon, J. Phys. Chem. A 102 (1998) 2050.
[14] G.E. Scuseria, C.A. Jiménez-Hoyos, T.M. Henderson, K. Samanta, J.K. Ellis, J. Chem. Phys. 135 (2011) 124108.
[15] For instance A. Bohr, B.R. Mottelson, Nuclear Structure, Vol. 1, Benjamin, 1969.
[16] C.L. Lawson, R.J. Hanson, D. Kincaid, F.T. Krogh, ACM Trans. Math. Software 5 (1979) 308;
J.J. Dongarra, J. Du Croz, S. Hammarling, R.J. Hanson, ACM Trans. Math. Software 14 (1988) 1;
J.J. Dongarra, J. Du Croz, S. Hammarling, R.J. Hanson, ACM Trans. Math. Software 14 (1988) 18;
J.J. Dongarra, J. Du Croz, I.S. Duff, S. Hammarling, ACM Trans. Math. Software 16 (1990) 1;
J.J. Dongarra, J. Du Croz, I.S. Duff, S. Hammarling, ACM Trans. Math. Software 16 (1990) 18.
[17] K.R. Wadleigh, I.L. Crawford, Software Optimization for High Performance Computing: Creating Faster Applications, Prentice Hall, 2000.
[18] V. Sarkar, W. Harrod, A.E. Snavely, J. Phys.: Conf. Ser. 180 (2009) 012045.
[19] D.J. Thouless, Nuclear Phys. 21 (1960) 225.
[20] G.C. Wick, Phys. Rev. 80 (1950) 268.